Tipos de Valor y Tipos de Referencia

Un tipo se define como un conjunto de datos y las operaciones realizadas en ellos. C# es un lenguaje fuertemente tipado y el sistema de tipo C# contiene tres categorías de tipo que son: tipos de valor, tipos de referencia y tipos de puntero. Los tipos de valor almacenan los datos, mientras que los tipos de referencia almacenan referencias a los datos reales. Los tipos de puntero variable se usan solo en modo inseguro (**unsafe mode**). Los tipos de valor derivan de **System.ValueType** y los tipos de referencia derivados de **System.Object**.

La principal diferencia entre los tipos de valor y los tipos de referencia es que estos tipos almacenan los valores en la memoria. El **Common Language Runtime (CLR)** asigna los valores en la memoria **Stack (ejecución LIFO)** y **Heap (almacenamiento libre/dinámico)**. Un tipo de valor mantiene su valor real asignado en la memoria Stack mientras que los tipos de referencia denominados objetos, almacenan las referencias a los datos reales. En C # es posible convertir un valor de un tipo a un valor de otro tipo. La operación de convertir un tipo de valor en un tipo de referencia se llama **Boxing** y la operación inversa se llama **Unboxing**.

Ejemplo Tipos de Valor

El contenido de una variable de tipo de valor es un valor.

Por ejemplo, el contenido del tipo de valor incorporado, int, es de 32 bits de datos.

La asignación de una instancia de tipo de valor siempre copia la instancia.

Por ejemplo:

```
using System;

struct Puntero { public int X, Y; }
class ClasePrincipal
{
    public static void Main(string[] args)
    {
        Puntero p1 = new Puntero();
        p1.X = 1;

        // la asignación genera una copia

        Puntero p2 = p1;

        Console.WriteLine(p1.X);
        Console.WriteLine(p2.X);

        // cambia p1.x

        p1.X = 2;
```

```
        Console.WriteLine(p1.X);
        Console.WriteLine(p2.X);
        Console.ReadKey();
    }

}
```

Resultado

```
1

1

2

1
```

Ejemplo Tipos de Referencia

Un tipo de referencia tiene dos partes: un objeto y su referencia.

El contenido de una variable de tipo de referencia es la referencia.

```
using System;

class Puntero
{
    public int X, Y;
}

public class ClasePrincipal
{
    public static void Main(string[] args)
    {
        Puntero p1 = new Puntero();
        p1.X = 1;

        // copia la referencia a p1

        Puntero p2 = p1;

        Console.WriteLine(p1.X);
        Console.WriteLine(p2.X);

        // cambia p1.x

        p1.X = 2;

        Console.WriteLine(p1.X);
        Console.WriteLine(p2.X);
        Console.ReadKey();
```

```
    }

}
```

Al asignar una variable de tipo de referencia a otra variable, se copia la referencia, no el objeto en sí. Es como realizar una copia de una llave. Puede usar múltiples variables para referirse al mismo objeto, de igual manera que puedes tener varias llaves de tu habitación.

Resultado

```
1

1

2

2
```

Boxing y Unboxing

El **boxing** es el proceso de convertir un tipo de valor en un tipo de referencia, que puede ser el objeto de la clase o una interfaz.

En el siguiente código hacemos **boxing** de un entero a un objeto:

```
    int numero = 1;

    object obj = numero;
```

Unboxing es la operación de devolver el objeto al tipo de valor original:

```
    int x = (numero)obj;
```

La operación de **Unboxing** requiere un casting explícito.

Si el tipo de valor no coincide con el tipo de objeto, C # lanzará una excepción del tipo **InvalidCastException**.

Por ejemplo, el siguiente código lanza una excepción, porque **long** no coincide exactamente con **int**:

```
    // aquí el valor 9 se entiende que es del tipo int
    object obj = 9;

    long x = (long)obj;
```

Tendríamos que realizar la siguiente modificación para que no provoque el error:

```
    object obj = 9;

    long x = (long)obj;
```

El siguiente código muestra cómo realizar una operación de **unboxing** de un valor doble a un valor entero.

```
    // se entiende que 5.5 es de tipo double
    object obj = 5.5;
    // x ahora es 5

    int x = (int)(double)obj;
```

FIN